KENYON & KENYON
Intellectual Property Law

333 W. San Carlos Street
Suite 600
San Jose, CA 95110-2731
408.975.7500
Fax 408.975.7501

Fax Transmission

From:	**Frank L. Bernstein**	Date:	November 15, 2005
Direct Dial:	408.975.7988	Fax:	408.975.7501
Docket Number:	SVL920010003US1 (13296/2)	Total number of pages: (including cover)	27

Please deliver to:

Name	Company	Fax	Phone
APPEAL BRIEF - PATENTS	**U.S. Patent and Trademark Office**	**571.273.8300**	

Message:

Application No.	:	09/990,802
Applicant	:	Eitan FARCHI et al.
Filed	:	November 13, 2001
Title	:	METHOD AND APPRATUS FOR COLLECTING PERSISTENT COVERAGE DATA ACROSS SOFTWARE VERSIONS
TC/A.U.	:	2124
Examiner	:	Jason D. MITCHELL

PAPER(s) ENTITLED:	Fee Transmittal (plus 1 copy)	2 pages
	Appeal Brief	24 pages

77453

☒ Original will not follow ☐ Original will follow by ☐ Regular Mail ☐ Overnight Delivery ☐ Hand Delivery

FEE TRANSMITTAL for FY 2005

Effective 10/01/2004. Patent fees are subject to annual revision.

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) **500.00**

Complete If Known	
Application Number	09/990,802
Filing Date	November 13, 2001
First Named Inventor	Eitan FARCHI et al.
Examiner Name	Jason D. Mitchell
Art Unit	2193
Attorney Docket No.	SVL920010003US1 (13296/2)

METHOD OF PAYMENT *(check all that apply)*

☐ Check ☐ Credit card ☐ Money Order ☐ Other ☐ None

☒ Deposit Account:

Deposit Account Number: 11-0600

Deposit Account Name: Kenyon & Kenyon

The Director is authorized to: *(check all that apply)*
☒ Charge fee(s) indicated below ☒ Credit any overpayments
☒ Charge any additional fee(s) or any underpayment of fee(s)
☒ Charge fee(s) indicated below, except for the filing fee to the above-identified deposit account.

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1001	790	2001	395	Utility filing fee	
1002	360	2002	175	Design filing fee	
1003	550	2003	275	Plant filing fee	
1004	790	2004	395	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	
				SUBTOTAL (1)	($) 0

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

	Extra Claims		Fee from below	Fee Paid
Total Claims	- 20 =		X 50.00 =	
Independent Claims	- 3 =		X 200.00 =	
Multiple Dependent			X =	

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
1202	50	2202	25	Claims in excess of 20
1201	200	2201	100	Independent claims in excess of 3
1203	360	2203	180	Multiple dependent claim, if not paid
1204	200	2204	100	** Reissue independent claims over original patent
1205	50	2205	25	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($)

**or number previously paid, if greater; For Reissues, see above*

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet.	
1053	130	1053	130	Non-English specification	
1812	2,520	1812	2,520	For filing a request for *ex parte* reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	120	2251	60	Extension for reply within first month	
1252	450	2252	225	Extension for reply within second month	
1253	1,020	2253	510	Extension for reply within third month	
1254	1,590	2254	795	Extension for reply within fourth month	
1255	2,160	2255	1,080	Extension for reply within fifth month	
1401	500	2401	250	Notice of Appeal	
1402	500	2402	250	Filing a brief in support of an appeal	500.00
1403	1,000	2403	500	Request for oral hearing	
1451	1,510	1451	1,510	Petition to institute a public use proceeding	
1452	500	2452	250	Petition to revive – unavoidable	
1453	1,500	2453	750	Petition to revive – unintentional	
1501	1,400	2501	685	Utility issue fee (or reissue)	
1502	490	2502	245	Design issue fee	
1503	660	2503	330	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17 (q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	
1809	790	2809	395	Filing a submission after final rejection (37 CFR § 1.129(a))	
1810	790	2810	395	For each additional invention to be examined (37 CFR § 1.129(b))	
1801	790	2801	395	Request for Continued Examination (RCE)	
1802	900	1802	900	Request for expedited examination of a design application	

Other fee (specify) ____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($) 500.00

SUBMITTED BY				Complete (if applicable)	
Name (Print/Type)	Frank L. Bernstein	Registration No. (Attorney/Agent)	31,484	Telephone	(408) 975-7988
Signature	Frank L. Bernstein			Date	November 15, 2005

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this paper is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on November 15, 2005.

Dilar Rodriguez

Patent **Attorney Docket No.: SVL920010003US1**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application No.	:	09/990,802	Confirmation No. 3720
Applicant	:	Eitan FARCHI	
Filed	:	November 13, 2001	
Title	:	METHOD AND APPARATUS FOR COLLECTING PERSISTENT COVERAGE DATA ACROSS SOFTWARE VERSIONS	
TC/A.U.	:	2124	
Examiner	:	Jason D. MITCHELL	

APPEAL BRIEF

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This brief is in furtherance of the Notice of Appeal, filed in this case on September 15, 2005.

TABLE OF CONTENTS

APPEAL BRIEF
U.S. Application No. 09/990,802

I. REAL PARTY IN INTEREST

The real party in interest in this matter is International Business Machines Corporation (*see* the Assignment document recorded on November 14, 2001 at Reel 012323, Frame 0185.).

II. RELATED APPEALS AND INTERFERENCES

There are neither related appeals nor related interferences affecting this application.

III. STATUS OF CLAIMS

Claims 1-21 are all the claims pending in this application.

Claims 1, 6-8, 13-15, and 20-21 are rejected under 35 U.S.C. §102(b) as being anticipated by USP 5,673,387 (Chen). Claims 2-3, 9-10, 16-17 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Chen in view of "Managing data through naming standards" by Winder, Software, IEEE, Vol. 7, Issue 4, July 1990 (Winder). Claims 4, 11, and 18 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Chen in view of USP 5,778,169 (Reinhardt). Claims 5, 12, and 19 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Chen.

IV. STATUS OF AMENDMENTS

Appellants did not file an Amendment in response to the Final Office Action of June 16, 2005.

The listing of claims beginning on page 16 of this Appeal Brief reflects the present status of the claims.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present invention relates to a method, apparatus, and article of manufacture for a computer-implemented system for collecting persistent code coverage data across software versions.

There have been many different approaches to address the problem of determining what code remains unchanged after development of a new version, to reduce testing requirements. For large programs, running an extensive test suite, much of it on unchanged code, is time-consuming and inefficient.

As discussed in the Background section of the present application, two of the references on which the Examiner relies, Chen and Reinhardt, take different approaches to solving this problem. Chen looks at what it calls code coverage entities – types, functions, variables, and macros – and compares changed entities to covered entities. Test units are rerun if a covered entity is identified as having been changed. Reinhardt generates a test coverage matrix to identify a subset of tests to be run, based on coverage points inserted into source code.

In all of these prior approaches, code coverage data is collected on all of the code, even if some code coverage data from prior versions is still valid. As a result, code coverage data collection still has contained inefficiencies.

In contrast to these and other prior approaches, the present invention keeps track of code coverage data that persists over various versions of a software program product. With this approach, the need to run an entire test suite is reduced or eliminated.

The retention of code coverage data across software versions is an aspect of the invention that distinguishes the invention from prior approaches. The present application uses the adjective "persistent" to identify code coverage elements that continue across versions.

Claim 1 of the present application, containing similar language to independent claims 8 and 15, uses the terms "persistent code coverage data" for such data. The invention also uses the term "code coverage tasks" and identifies code coverage tasks by a persistent unique name.

The independent claims of the present application generally track the flow chart of Fig. 2, which is discussed at pages 12-16 of the application. Dependent claims 6, 13, and 20 generally track the flow chart of Fig. 5, which is discussed at 22-24 of the application.

Figure 2 discloses one embodiment in which, in step 206, code coverage tasks receive unique names so that if a code block is unchanged, it is not necessary to collect code coverage data for that task. This naming approach removes the problem of a code coverage task appearing to have changed if its absolute location is changed (see page 13 of the application). At the top of page 14, it is noted that other naming conventions could accomplish the same goal. The idea is to avoid regeneration of code coverage task names just because unrelated code lines have changed.

Another aspect of the invention, illustrated by way of example in Figure 5 and recited in claim 6 which depends from claim 1, and also in claim 13 (depending from claim 8) and claim 20 (depending from claim 15), relates to assigning new persistent unique names to code coverage tasks that do change across versions.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1, 6-8, 13-15, and 20-21 are rejected under 35 U.S.C. §102(b) as being anticipated by USP 5,673,387 (Chen).

2. Claims 2-3, 9-10, 16-17 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Chen in view of "Managing data through naming standards" by Winder, Software, IEEE, Vol. 7, Issue 4, July 1990 (Winder).

3. Claims 4, 11, and 18 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Chen in view of USP 5,778,169 (Reinhardt).

4. Claims 5, 12, and 19 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Chen.

VII. ARGUMENT

Rejections 1, 3, and 4

Appellants believe that the Examiner has misread the claimed invention and the prior art in three basic ways.

"Persistent"

First, the Examiner effectively has read out the word "persistent" from the claims. Appellants have used that term to define code coverage tasks that persist across software versions. Without that term, it is indeed possible that one could view the claimed invention as merely another attempt to gather code coverage data, and thus read the claims on various prior art. However, the assignment of a persistent unique name to a code coverage task is simply not something that the prior art does in any fashion.

"Unique Name"

Looking more closely at this term, and in particular at the Examiner's attempt (final Office Action, page 6) to read "persistent unique name" on portions of Chen (col. 9, lines 1-3 and col. 11, lines 20-21), what Chen is doing is determining whether an "entity" (a function, type, variable, or macro) in a newer version of a program matches an entity in a prior version. As Chen discusses at col. 10, lines 13-55, once two entities from respective versions are determined to match, it is not the entity name that determines whether the code in the entity has changed. Rather, it is a comparison of the checksum attributes of the respective entities that enables that determination.

The checksum attributes that Chen discusses are significant. In the present application, Appellants discuss the problem of unchanged code having changed absolute locations and the

like (see the paragraph bridging pages 13 and 14 of the present application). Chen's approach requires taking this change in absolute location into account. With the present invention, absolute location can be effectively ignored.

The "name" that Chen refers to is a name that is assigned to an entity when that entity is created as part of the code. The Chen "name" is not something that is assigned as part of code coverage data generation.

Therefore, Appellants submit that the Examiner's reading of the claims on Chen breaks down at this point. There is simply nothing in Chen, nor in any of the other prior art of record, that teaches or suggests the collection of persistent code coverage data, involving *inter alia* the assignment of persistent unique names to code coverage tasks. Consequently, Appellants submit that all of claims 1-21 of the present application are patentable.

"Code Coverage Task"

On pages 12 and 13 of the application, in the course of describing Figure 2 and steps 202 and 204, Appellants define the term "code coverage task" as follows:

> A code coverage task is a basic block of code for which an execution of a test returns a true value if the testing requirement of the task is fulfilled and a false value if the testing requirement of the task is not fulfilled. A basic block is a set of consecutive statements with a single entry point (i.e. the first statement) and a single exit point (i.e. the last statement). Control statements such as the "if" statement are considered as a separate block to ease the detection of source code changes that affect the associated blocks (i.e. basic blocks which follow the control statement). Source code changes will be discussed in more detail later. Those of ordinary skill in the art will recognize that there are other alternative

ways to divide a program source code into coverage tasks. For example, coverage tasks could be at module level, block level or statement level and could be identified manually rather than automatically and could be based on the user's needs.

(emphasis added)

In reading the claims of the present application on Chen, the Examiner has pointed to Chen's reference to "basic code entities" at col. 2, lines 47-50. At col. 8, beginning at line 8, Chen discusses four kinds of entities in a C program: types 180, functions 182, variables 184, and macros 188. At col. 3, beginning at approximately line 46, Chen describes two sets of entities in a software system S: functions F and nonfunctions V. Chen states, "functions are the basic entities that execute program semantics by creating and storing values. It is assumed that every action of a program must be carried out in some function. Nonfunctions are nonexecuting entities in a program such as variables, types, and macros. For example, variables define storage areas that functions manipulate, and types, among other things, define the storage extent of variables. A program in the software system is defined as a composition of some subsets of functions F and nonfunctions V."

From the foregoing, it can be appreciated that, while the Examiner has attempted to read the claimed "code coverage tasks" on Chen's "basic code entities," the two things are quite different from each other. Actually, the claimed "code coverage tasks" correspond to a Chen program, or to a block of a Chen program, and not to a function, variable, type, or macro.

The Examiner has pointed to the general language in the paragraph bridging pages 12 and 13 of the application ("Those of ordinary skill in the art will recognize that there are other alternative ways to divide a program source code into coverage tasks") as reading on Chen.

However, the Examiner's expansive reading of this language ignores the following sentence from that quote which gives an example of the kind of "other alternative ways" that the inventors contemplated: "For example, coverage tasks could be at module level, block level or statement level and could be identified manually rather than automatically and could be based on the user's needs."

Thus, the claimed code coverage tasks are very different from Chen's "entities". Once this distinction is appreciated, numerous recitations in the claims fall away from Chen. For example, looking at independent claim 1, "dividing the program source code statements of said computer program into a plurality of code coverage tasks" is something that Chen does not do with its entities, contrary to the Examiner's assertions. The difference pertains further when it comes to the generation of a persistent unique name for code coverage tasks. There is a tremendous difference between giving a persistent unique name to a code block, for purposes of having it left out of test suite running, and naming a function.

Based on the foregoing, when the present invention inserts coverage points into the computer program source code for each of the code coverage tasks, as claimed for example in claim 1, the portion of Chen on which the Examiner relies at col. 7, lines 7-9 referring to the adding of instrumentation, has nothing to do even with the Chen "entities" which the Examiner is attempting to the claimed code coverage tasks.

Looking further at claim 1, the creation of a code coverage database using the code coverage tasks in no way corresponds to Chen's generation of an entity trace list for each test unit, to which the Examiner refers at col. 9, lines 32-35 of Chen.

Pursuant to the foregoing, Appellants submit that independent claim 1, as well as independent claim 8, and independent claim 15, which contain corresponding recitation, are

patentable, as are all of the dependencies of these claims, including claims 4, 5, 11, 12, 18, and 19, which are the subject of rejections 3 and 4 as set forth in the preceding section. Therefore, Appellants submit that claims 1-21 in the subject application are patentable.

Looking also now at dependent claims 6, 13, and 20, similar rationales as with respect to the independent claims 1, 8, and 15 applies. That is, the use of the term "persistent"; the generation of "persistent unique names" for new and modified code coverage tasks; and the lack of correspondence between Chen's "entities" and the claimed "code coverage tasks" prevents any kind of reading of these dependent claims 6, 13, and 20 on Chen. Therefore, Appellants submit that these claims are patentable for this additional reason as well, as are their corresponding dependencies, claims 7, 14, and 21.

Rejection 2

Neither Winder nor Reinhardt supplies the deficiencies of Chen. The Examiner relies on Winder because of some general, highly non-specific statements therein about naming conventions. However, from the foregoing discussion, it should be appreciated that combining Winder with Chen would result in most at something on the order of having a unique name for a function in some source code. Winder would appear to serve no purpose, then, as having a unique name for a function would be an indispensable part of writing source code, separate and apart from any issue of code coverage issues in testing modified source code. Therefore, concerning claims 2, 9, and 16, since Chen's entities do not correspond to the claimed code coverage tasks, the unique naming in these claims finds no response in Chen, as the Examiner acknowledges, but also finds no response in Winder because there is simply no motivation to make any kind of suggestions (assuming for the sake of argument that there are suggestions) in Winder to modify the teachings of Chen – either Chen inherently provides unique names for

functions, or applying Winder has nothing to do with code coverage issues. In any event, such modification would not yield the subject matter of claims 2, 9, and 16 because Chen's entities in no way correspond to the claimed code coverage tasks. Appellants note further that, for example, a statement such as, "your filing system determines your ability to access the information you need to manage," as Winder states in the first full paragraph in col. 3 on page 84, provides no suggestion whatsoever to the ordinarily skilled artisan as to any details of implementation, nor as to what should be done to Chen to modify it.

Likewise, concerning claims 3, 10, and 17, Appellants submit that the Examiner cannot: 1) take some general reference to a three-part naming convention in Winder; 2) acknowledge that it is different from the naming convention in the present invention; 3) combine that different naming convention with something different again in Chen, and 4) somehow make a leap to arrive at the naming set forth in these claims. Again, the lack of correspondence between Chen's entities and the claimed code coverage tasks prevents, among other things, the link the Examiner is trying to make here. Therefore, Appellants submit that claims 2, 3, 9, 10, 16, and 17 are patentable for these additional reasons as well.

Pursuant to the foregoing, Appellants submit that all of claims 1-21 in the subject application are patentable.

The Examiner is hereby authorized to charge the appeal brief fee of **$500.00** and any additional fees which may be necessary for consideration of this paper to Kenyon & Kenyon Deposit Account No. **11-0600.**

Respectfully submitted,

KENYON & KENYON

APPEAL BRIEF
U.S. Application No. 09/990,802

Date: November 15, 2005

By: Frank L Bernstein
Frank L. Bernstein
Reg. No. 31,484

CLIENT NO. **25693**

KENYON & KENYON
333 West San Carlos St., Suite 600
San Jose, CA 95110
Telephone: (408) 975-7500
Facsimile: (408) 975-7501

CLAIMS APPENDIX

1. A method using a computer system for collecting persistent code coverage data for a computer program, the computer program comprising program source code statements, the method comprising the steps of:

identifying the computer program for which the persistent code coverage data should be collected;

dividing the program source code statements of said computer program into a plurality of code coverage tasks;

generating a persistent unique name for each of the code coverage tasks of said plurality of code coverage tasks;

inserting coverage points into the computer program source code for each of the code coverage tasks to produce an instrumented program;

compiling and linking the instrumented program into a program executable;

identifying a set of test cases from a plurality of test cases to be run for the code coverage data collection purposes;

creating a code coverage database using the code coverage tasks and the identified set of test cases;

running the program executable with a test case from the identified set of test cases and writing the information about the test case and the coverage points that are executed into an output file, until all the test cases have been run; and

processing the information contained in the output file into code coverage data and populating the code coverage database with said code coverage data.

2. The method of Claim 1, wherein generating the persistent unique name for each of the code coverage tasks is done using a unique naming convention.

3. The method of Claim 2, wherein the unique naming convention comprises a computer program module name, a version indicator, and a unique code coverage task identifier.

4. The method of Claim 1, wherein the code coverage database comprises a table, the table comprising a row for each test case in said identified set of test cases and a column for

each code coverage task of said plurality of code coverage tasks, said column comprising an indicator at each row indicating coverage status for said code coverage task.

5. The method of Claim 1, wherein said computer program comprises program source code statements written in a hardware description language.

6. The method of Claim 1 further comprising the steps of:

modifying the computer program to produce a modified version of the computer program source code;

identifying a plurality of new, modified, and deleted code coverage tasks in said modified version of the computer program source code;

generating a persistent unique name for each of the new and modified code coverage tasks of said plurality of new, modified and deleted code coverage tasks;

inserting coverage points into the modified version of the computer program source code for each of the new and modified code coverage tasks to produce an instrumented modified version of the computer program source code;

compiling and linking the instrumented modified version of the computer program source code into a modified program executable;

identifying a new set of test cases from a plurality of test cases to be run for the code coverage data collection purposes on the new and modified code coverage tasks;

altering the code coverage database to accommodate new, modified and deleted code coverage tasks and the new set of test cases, and clearing any code coverage data for the modified code coverage tasks from said code coverage database;

running the modified program executable with a test case from the identified new set of test cases and collecting code coverage data for the new and modified code coverage tasks, until all the test cases have been run; and

updating the code coverage database with the collected code coverage data for the new and modified code coverage tasks;

whereby the previously collected code coverage data for the non-affected code coverage tasks is preserved from a previous version of the computer program to the modified version of said computer program eliminating the need for running the entire test bucket.

7. The method of Claim 6, wherein generating a persistent unique name for each of the modified code coverage tasks is done by changing the version indicator in the names of said modified code coverage tasks.

8. An apparatus for collecting persistent code coverage data for a program, the persistent code coverage data being stored in a code coverage database associated with the program, the program comprising program source code statements, the apparatus comprising:

a computer system having a data storage device connected thereto, wherein the data storage device stores the code coverage database; and

one or more computer programs executed by the computer system for:

identifying the program for which the code coverage data should be collected;

dividing the program source code statements of said program into a plurality of code coverage tasks;

generating a persistent unique name for each of the code coverage tasks of said plurality of code coverage tasks;

inserting coverage points into the program source code for each of the code coverage tasks to produce an instrumented program;

compiling and linking the instrumented program into a program executable;

identifying a set of test cases from a plurality of test cases to be run for the code coverage data collection purposes;

creating a code coverage database using the code coverage tasks and the identified set of test cases;

running the program executable with a test case from the identified set of test cases and writing the information about the test case and the coverage points that are executed into an output file, until all the test cases have been run; and

processing the information contained in the output file into code coverage data and populating the code coverage database with said code coverage data.

9. The apparatus according to Claim 8, wherein generating a persistent unique name for each of the code coverage tasks is done using a unique naming convention.

10. The apparatus according to Claim 9, wherein the unique naming convention comprises a program module name, a version indicator, a unique code coverage task identifier.

11. The apparatus according to Claim 8, wherein the code coverage database comprises a table, the table comprising a row for each test case in said identified set of test cases and a column for each code coverage task of said plurality of code coverage tasks, said column comprising an indicator at each row indicating coverage status for said code coverage task.

12. The apparatus according to Claim 8, wherein said program comprises program source code statements written in a hardware description language.

13. The apparatus according to Claim 8, wherein the one or more computer programs performed by the computer system further provides for:

modifying the program to produce a modified version of the program source code;

identifying a plurality of new, modified and deleted code coverage tasks in said modified version of the program source code;

generating a persistent unique name for each of the new and modified code coverage tasks of said plurality of new, modified and deleted code coverage tasks;

inserting coverage points into the modified version of the program source code for each of the new and modified code coverage tasks to produce an instrumented modified version of the program source code;

compiling and linking the instrumented modified version of the program source code into a modified program executable;

identifying a new set of test cases from a plurality of test cases to be run for the code coverage data collection purposes on the new and modified code coverage tasks;

altering the code coverage database to accommodate new, modified and deleted code coverage tasks and the new set of test cases, and clearing any code coverage data for the modified code coverage tasks from said code coverage database;

running the modified program executable with a test case from the identified new set of test cases and collecting code coverage data for the new and modified code coverage tasks, until all the test cases have been run; and

updating the code coverage database with the collected code coverage data for the new and modified code coverage tasks;

whereby the previously collected code coverage data for the non-affected code coverage tasks is preserved from a previous version of the program to the modified version of said program eliminating the need for running the entire test bucket.

14. The apparatus according to Claim 13, wherein generating a persistent unique name for each of the modified code coverage tasks is done by changing the version indicator in the names of said modified code coverage tasks.

15. An article of manufacture comprising a program storage device readable by a computer and tangibly embodying one or more programs of instructions executable by the computer to perform method steps for collecting persistent code coverage data for a computer program, the computer program comprising program source code statements, the method comprising the steps of:

identifying the computer program for which the code coverage data should be collected;

dividing the program source code statements of said computer program into a plurality of code coverage tasks;

generating a persistent unique name for each of the code coverage tasks of said plurality of code coverage tasks;

inserting coverage points into the computer program source code for each of the code coverage tasks to produce an instrumented program;

compiling and linking the instrumented program into a program executable;

identifying a set of test cases from a plurality of test cases to be run for the code coverage data collection purposes;

creating a code coverage database using the code coverage tasks and the identified set of test cases;

running the program executable with a test case from the identified set of test cases and writing the information about the test case and the coverage points that are executed into an output file, until all the test cases have been run; and

processing the information contained in the output file into code coverage data and populating the code coverage database with said code coverage data.

16. The article of manufacture according to Claim 15, wherein generating a persistent unique name for each of the code coverage tasks is done using a unique naming convention.

17. The article of manufacture according to Claim 16, wherein the unique naming convention comprises a computer program module name, a version indicator, a unique code coverage task identifier.

18. The article of manufacture according to Claim 15, wherein the code coverage database comprises a table, the table comprising a row for each test case in said identified set of test cases and a column for each code coverage task of said plurality of code coverage tasks, said column comprising an indicator at each row indicating coverage status for said code coverage task.

19. The article of manufacture according to Claim 15, wherein said computer program comprising program source code statements written in a hardware description language.

20. The article of manufacture according to Claim 15 further comprising the steps of:

modifying the computer program to produce a modified version of the computer program source code;

identifying a plurality of new, modified and deleted code coverage tasks in said modified version of the computer program source code;

generating a persistent unique name for the new and modified code coverage tasks of said plurality of new, modified and deleted code coverage tasks;

inserting coverage points into the modified version of the computer program source code for each of the new and modified code coverage tasks to produce an instrumented modified version of the computer program source code;

compiling and linking the instrumented modified version of the computer program source code into a modified program executable;

identifying a new set of test cases from a plurality of test cases that should be run for the code coverage data collection purposes on the new and modified code coverage tasks;

altering the code coverage database to accommodate new, modified and deleted code coverage tasks and the new set of test cases, and clearing any code coverage data for the modified code coverage tasks from said code coverage database;

running the modified program executable with a test case from the identified new set of test cases and collecting code coverage data for the new and modified code coverage tasks, until all the test cases have been run; and

updating the code coverage database with the collected code coverage data for the new and modified code coverage tasks;

whereby the previously collected code coverage data for the non-affected code coverage tasks is preserved from a previous version of the computer program to the modified version of said computer program eliminating the need for running the entire test bucket.

21. The article of manufacture according to Claim 20, wherein generating persistent unique name for the modified code coverage tasks is done by changing the version indicator in the names of said modified code coverage tasks.

EVIDENCE APPENDIX

No evidence is being entered nor relied upon in this Appeal.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.

SJ01 75332 v1